

# Biowulf: A Beowulf for Bioscience

Steven Fellini

[sfellini@nih.gov](mailto:sfellini@nih.gov)

Susan Chacko

[susanc@helix.nih.gov](mailto:susanc@helix.nih.gov)

CIT  
November 8, 2012

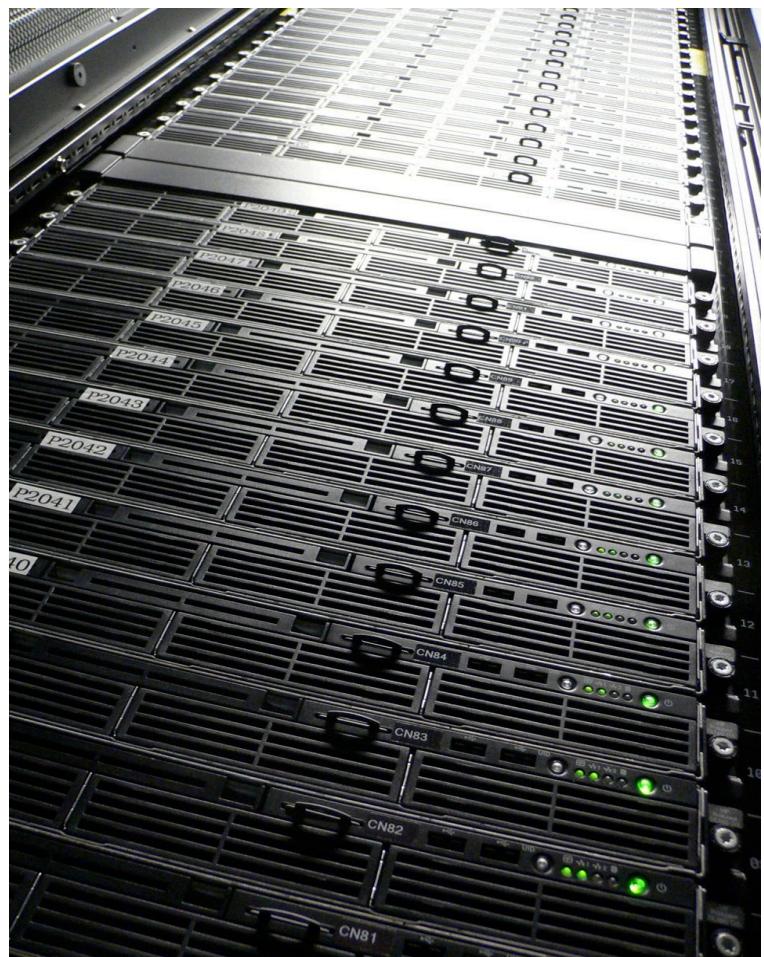
Slides available as <http://biowulf.nih.gov/biowulf-seminar-nov2012.pdf>

# Introduction to Biowulf

- Intro to the Biowulf NIH Cluster
- Cluster Basics & Concepts
- Biowulf architecture & hardware configuration
- Why use Clusters/Biowulf?
- Accounts & Passwords
- Connecting to Biowulf; Email
- Storage
- Running Jobs & the Batch System
- Monitoring Jobs
- Node allocation & selection
- Tour of Biowulf Machine Room

# The NIH Biowulf Cluster

- Central scientific compute resource managed by CIT Helix Staff
- Funded by NIH Management Fund
- Available to all NIH intramural scientists
- Production facility: high availability, data integrity
- Large scale: ~15,000 processor cores; Petabyte storage capacity
- Enabling research not otherwise possible
- General purpose scientific computing (not dedicated to any one application type)
- Cited by over 400 publications in 2009-2012





<http://biowulf.nih.gov>



# biowulf

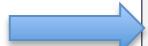
AT THE NIH

[Status](#)  
[Applications](#)  
[Hardware](#)  
[Storage](#)  
[User Guide](#)  
[Performance](#)  
[Research](#)  
[Photos](#)  
[About](#)



The NIH Biowulf cluster is a GNU/Linux parallel processing system designed and built at the National Institutes of Health and managed by the Helix Systems Staff. The system is designed for large numbers of simultaneous jobs common in bioinformatics as well as large-scale distributed memory tasks such as molecular dynamics.

<http://biowulf.nih.gov/research.html>



## Recent Publications:

(All collected publications)



[β-Barrel Topology of Alzheimer's β-Amyloid Ion Channels](#)  
 Hyunbum Jang, Fernando Teran Arce, Srinivasan Ramachandran, Ricardo Capone, Ratnesh Lal, and Ruth Nussinov  
*J. Mol. Biol.*, doi:10.1016/j.jmb.2010.10.025 (2010)



[Quantitative protein and mRNA profiling shows selective post-transcriptional control of protein expression by vasopressin in kidney cells](#)  
 Sookkasem Khositseth, Trairak Pisitkun, Dane H. Slentz, Guanghui Wang, Jason D. Hoffert, Mark A. Knepper\* and Ming-Jiun Yu  
*Molecular & Cellular Proteomics*, doi: 10.1074/mcp.M110.004036 (2010)

[An approach for jointly modeling multivariate longitudinal measurements and discrete time-to-event data](#)  
 Paul S. Albert and Joanna H. Shih  
*Ann. Appl. Stat.* 4(3) :1517-1532 (2010)

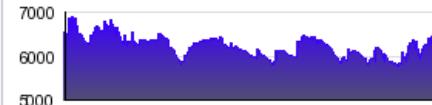


[Biological Validation of Increased Schizophrenia Risk With NRG1, ERBB4, and AKT1 Epistasis via Functional Neuroimaging in Healthy Controls](#)  
 Kristin K. Nicodemus, Amanda J. Law, Eugenia Radulescu, Augustin Luna, Bhaskar Kolachana, Radhakrishna Vakkalanka, Dan Rujescu, Ina Giegling, Richard E. Straub, Kate McGee, Bert Gold, Michael Dean, Pierandrea Muglia, Joseph H. Callicott, Hao-Yang Tan, Daniel R. Weinberger  
*Arch. Gen. Psych.* 67(10) :991-1001 (2010)

## Current Cluster Status

Thursday, October 28th, 2010

Biowulf load history, last 24 hours



In the last hour:

166 jobs started  
249 jobs finished

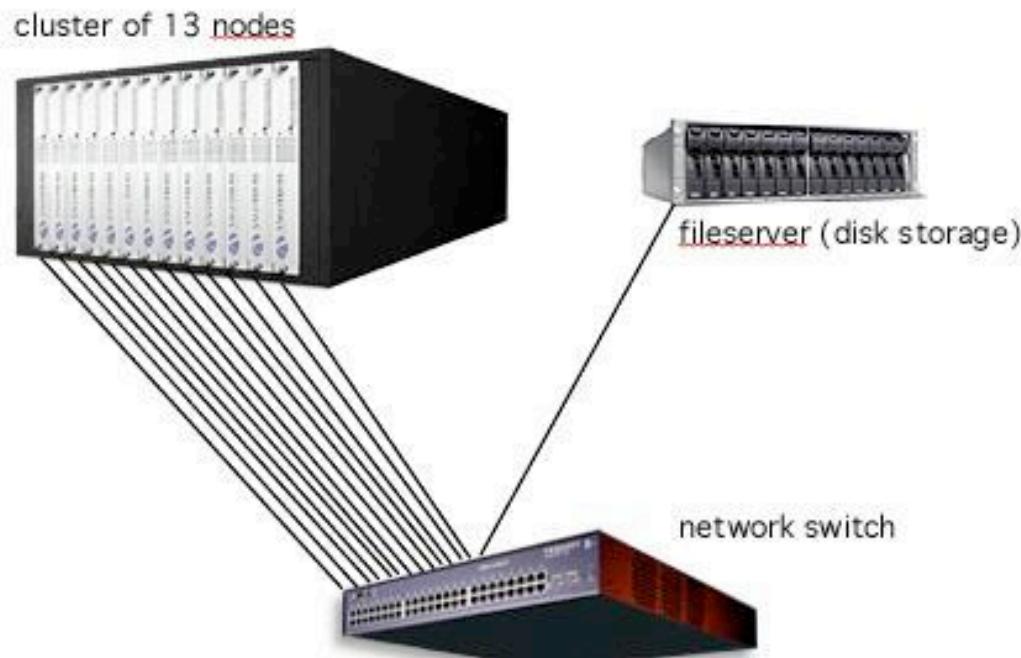
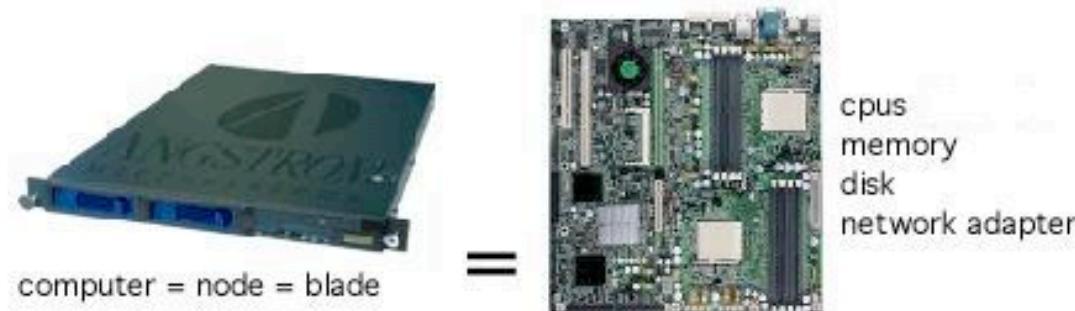
Users and Jobs:  
88 users  
1004 running jobs

## Recent News:

(All news)

- [Use of large memory compute nodes on NIH Biowulf](#) (Oct 13th 2010)
- [NIH Biowulf seminar Nov 4](#) (Oct 13th 2010)
- [Faster file transfers on Biowulf](#) (Oct 4th 2010)

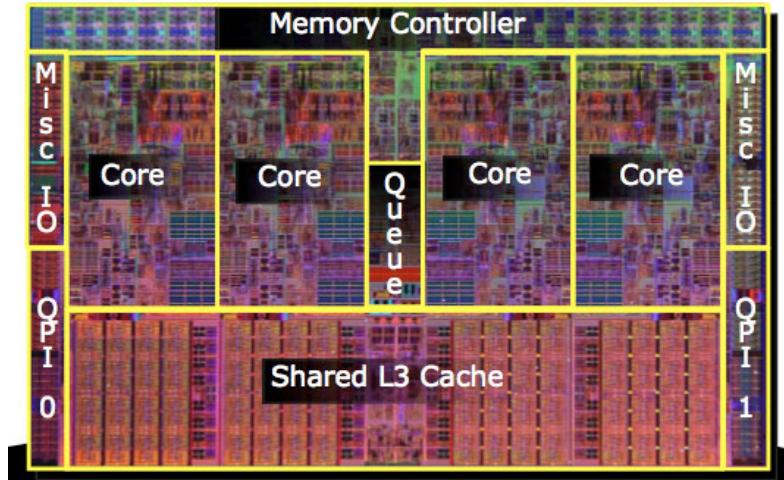
# Cluster Basics



# Terminology/Concepts

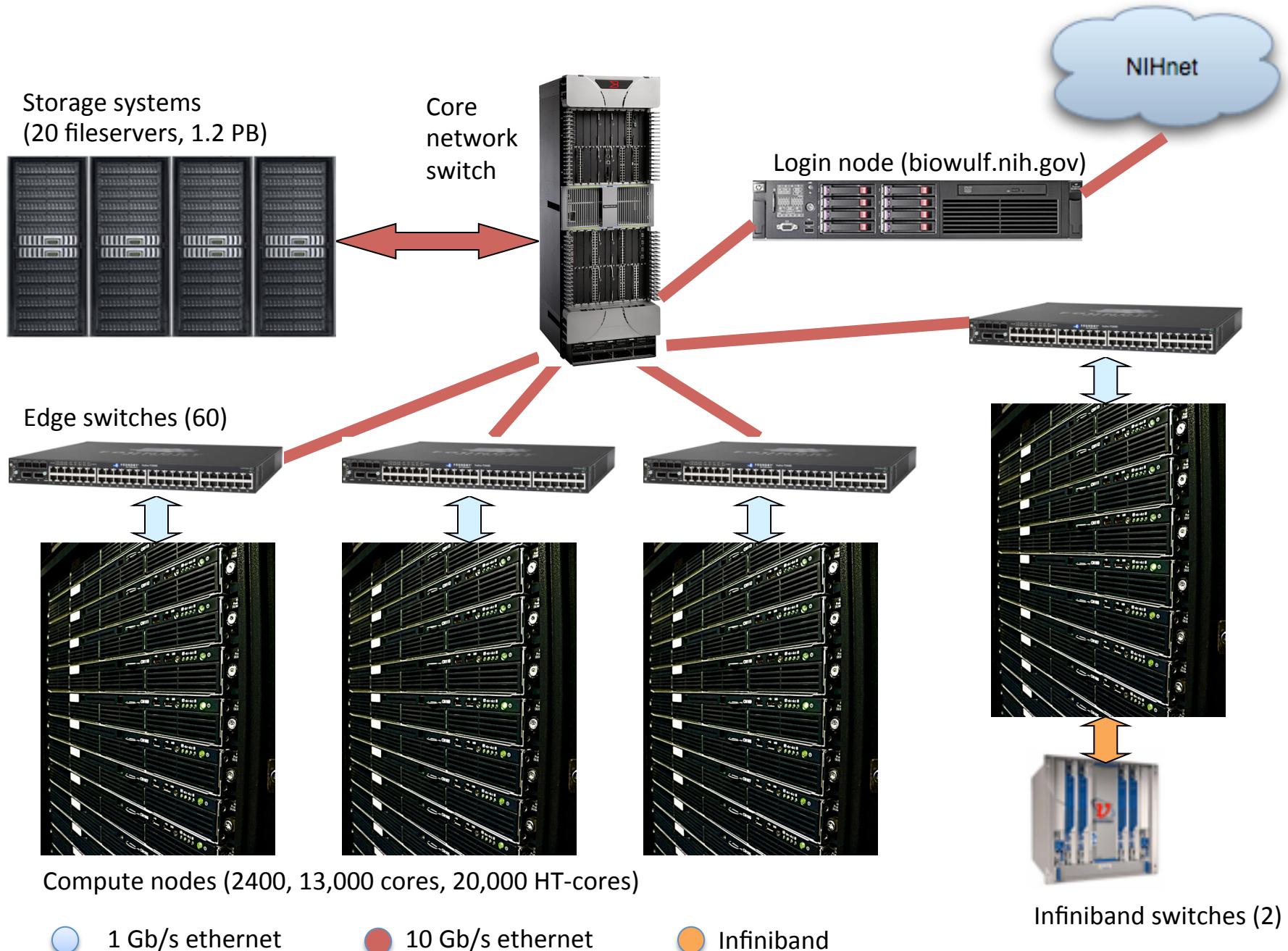


What should we call this?



- Node
- Processor (or CPU) vs. Core
- Hyper-threading, hyper-threaded cores
- Process vs. Thread
- Objective: 1 thread => 1 core

# NIH Biowulf Cluster Architecture



# Compute Node Technical Specifications

# nodes	cores per node	Memory	network
384	12 x 2.8 GHz Intel Xeon (X5660) 12 MB secondary cache <a href="#">Hyper-threading enabled</a>	24 GB	1 Gb/s ethernet
352	8 x 2.67 GHz Intel Xeon (X5550) 8 MB secondary cache <a href="#">Hyper-threading enabled</a>	320 x 24 GB 32 x 72 GB	1 Gb/s ethernet
1	32 x 2.5 GHz AMD Opteron 8380 512 KB secondary cache	512 GB	1 Gb/s ethernet
250	8 x 2.8 GHz Intel Xeon (E5462) 12 MB secondary cache	8 GB	1 Gb/s ethernet 16 Gb/s infiniband
232	4 x 2.8 GHz AMD Opteron 290 1 MB secondary cache	8 GB	1 Gb/s ethernet
289	4 x 2.6 GHz AMD Opteron 285 1 MB secondary cache	8 GB	1 Gb/s ethernet
471	2 x 2.8 GHz AMD Opteron 254 1 MB secondary cache	40 x 8 GB 226 x 4 GB 91 x 2 GB	1 Gb/s ethernet 160 x 8 Gb/s infiniband
389	2 x 2.2 GHz AMD Opteron 248 1 MB secondary cache	129 x 2 GB 66 x 4 GB	1 Gb/s ethernet
91	2 x 2.0 GHz AMD Opteron 246 1 MB secondary cache	48 x 1 GB 43 x 2 GB	1 Gb/s ethernet

# Coming in December...

- 24 x 256 GB nodes (16-core, 32-hypercore)
- NIDDK-funded: 206 Infiniband-connected nodes (32 & 64 GB, 16-core, 32-hypercore)

# Compute Node Configurations, continued

- Heterogeneous: 15 node configurations
- 2-4 GB swap
- 60-200 GB local scratch disk
- RedHat/CentOS 5.x, 64-bit

# Why would you want to use Biowulf?

- Large numbers of jobs (bioinformatics on thousands of sequences)
- Large-memory jobs (genome assemblies)
- Parallel jobs with high-cpu demands (e.g., molecular dynamics on 256 cores)
- Terabytes of data to process

... in other words, LARGE SCALE

# Unsuitable for Biowulf...(or, why bother?)

- Small numbers of jobs
- One dependent job after the other
- Interactive jobs, graphics etc.

... in other words, desktops can be pretty powerful! (Helix too)

# Class survey

- Level of Unix/Linux experience?
- Helix account?
- Biowulf account?
- Your research application?

# Accounts & Passwords

- Every user must have his/her own account. NO SHARING of accounts
- Requires a pre-existing Helix account (see <http://helix.nih.gov/Documentation/accounts.html>)
- Registering for a Biowulf account (see [https://helix.nih.gov/nih/account\\_request.html](https://helix.nih.gov/nih/account_request.html))
- Passwords – NIH Login (AD) password
- Default login shell: bash
- No \$ charges associated with your Biowulf account (/home directory usage is a Helix account charge)

# Connecting to Biowulf & Email

- ssh to biowulf.nih.gov (see [http://helix.nih.gov/new\\_users/connect.html](http://helix.nih.gov/new_users/connect.html))
- Must be on NIH VPN (or login from helix)
- Email goes to [your username@helix.nih.gov](mailto:your_username@helix.nih.gov)
- When you apply for a biowulf account:

## Email Address:

I will read my email from Biowulf systems and staff at my Helix address:

Yes  No, my preferred email address is:

- **Important:** make sure your helix email is being properly forwarded if you don't normally read it (see <http://helix.nih.gov/Email/>)

# Transferring files

<http://helix.nih.gov/Documentation/transfer.html>

- Map your Helix /home or /data area on your desktop.
- GUI File transfer clients (WinSCP etc.)
- Commandline (scp, sftp)
- scp-hpn (<http://helix.nih.gov/News/?314>)
- Remember: Windows files have Ctrl-M characters (dos2unix)

# Use of Biowulf's login node

- Submitting jobs
- Editing/compiling code
- File management
- File transfer
- **Brief testing of code or debugging (under 20 minutes cpu time)**
- **Do not run compute jobs of any kind on the login node!**

# Disk storage

	<b>Location</b>	<b>Creation</b>	<b>Backups?</b>	<b>Amount of space</b>	<b>Accessible from (*)</b>
/home	fileservers	with Helix account	Yes	8 GB (quota)	B, H, C
/data	fileservers	with Biowulf account	Yes	100 GB (quota)	B, H, C
/scratch	fileservers	created by user	No	1000 GB shared by all users	B, H
/scratch	local disk	created by user	No	varies up to 200 GB, dedicated while node is allocated	C

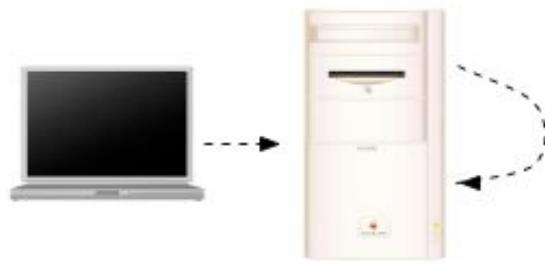
(\*) H = helix, B = biowulf login node, C = biowulf computational nodes

# Disk Storage

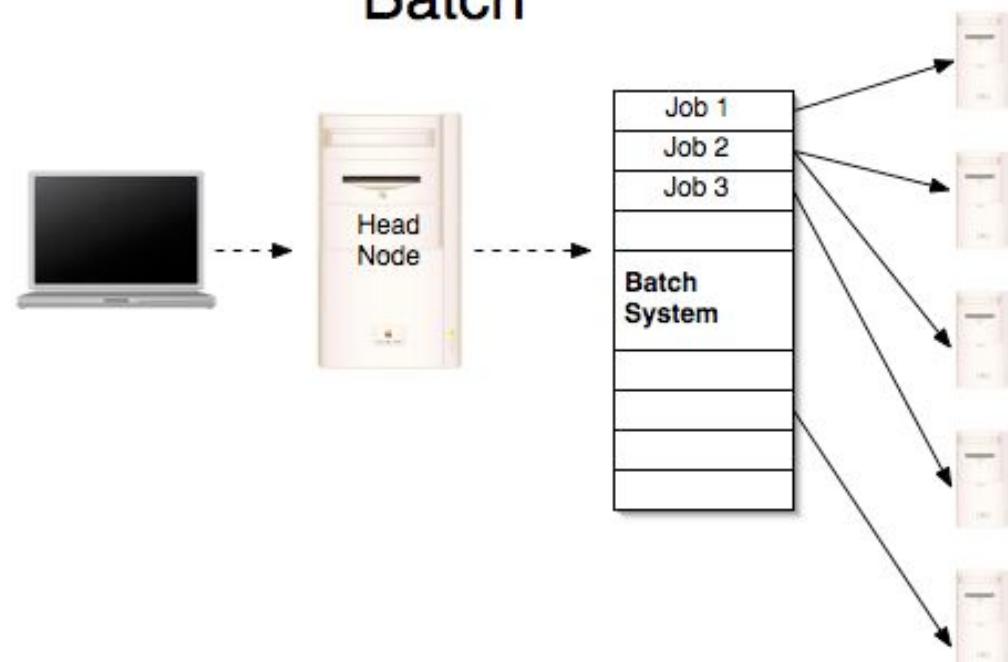
- Use /data/username, **not** /gs1/users/username
- *checkquota* command to check usage
- *clearscratch* command to clear out local /scratch on compute nodes
- Login node: use /scratch, **not** /tmp or /var/tmp
- Quota increases for /data – request at  
[https://helix.nih.gov/nih/storage\\_request.html](https://helix.nih.gov/nih/storage_request.html)
- access from your workstation (see  
<http://helix.nih.gov/Documentation/transfer.html>)
- Snapshots – 6 daily, 2 nightly, 4 weekly on /home directories (less frequent on /data)

# Concepts: Interactive vs. Batch

Interactive

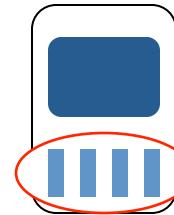
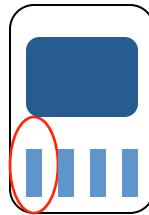


Batch

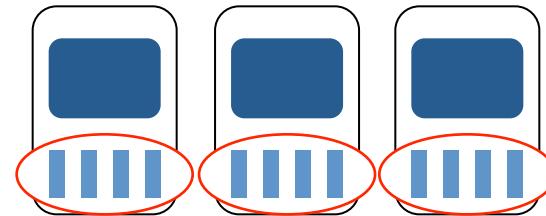
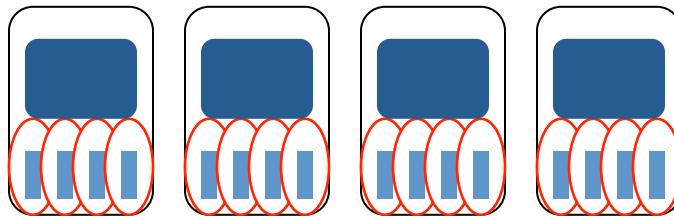


# Kinds of Jobs on a Cluster

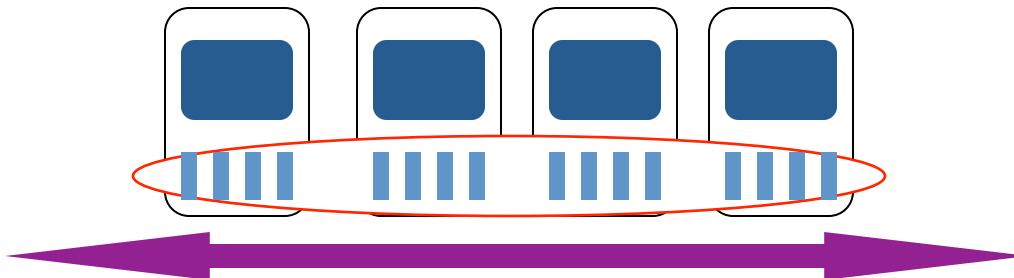
- Single-threaded apps
- Shared-memory parallel apps (multi-threaded)



- Swarms



- Distributed-memory parallel apps



# Jobs on the Biowulf Cluster

- PBS is the batch system (queuing software)
- Unit of allocation = node (2, 4, 8, 16 or 24 cores)
- Nodes are *dedicated* during the course of the job
- Each allocated core should run a single thread (exception = per process memory requirements)
- Kinds of Jobs: serial, parallel, multi-threaded, and *large memory*
- Most (almost all) jobs should be run in batch
- There is a maximum core allocation; all cores in a node are charged to you whether you use them or not

# Submitting a simple serial job

```
#!/bin/bash
#
# this file is myjob.sh
#
#PBS -N MyJob
#PBS -m be
#
myprog -n 100 < infile
```

```
% qsub -l nodes=1 myjob.sh
```

- `-l nodes=1` is required (even for single node jobs)
- `myprog` output will appear in `MyJob.oNNNNNN` (STDIN) and `MyJob.eNNNNNN` (STDOUT)

# Essential PBS Commands

- qsub ...
- qdel [-Wforce] <jobid>
- qstat -a | -u <user> [-r] [-n] | -f <jobid>
- qselect -u <user> [-s R|Q]

# Serial Jobs, continued

What's inefficient about the previous example?

```
#!/bin/bash
#
# this file is myjob2.sh
#
#PBS -N MyJob
#PBS -m be
#
myprog -a 100 < infile1 > outfile1 &
myprog -a 200 < infile2 > outfile2 &
wait
```

# Submitting swarms of jobs with *swarm*

```
#  
# this file is cmdfile  
#  
myprog -param a < infile-a > outfile-a  
myprog -param b < infile-b > outfile-b  
myprog -param c < infile-c > outfile-c  
myprog -param d < infile-d > outfile-d  
myprog -param e < infile-e > outfile-e  
myprog -param f < infile-f > outfile-f  
myprog -param g < infile-g > outfile-g
```

```
% swarm -f cmdfile
```

# swarms

- *swarm* will place your jobs on 24, 16, 4, 2-core nodes depending on fit and availability
- 600-line file (600 threads) => 25 jobs (24-core) or 38 jobs (16-core) or 150 jobs (4-core) or 300 jobs (2-core)
- Entire swarm will be placed on one kind of node only
- Use semi-colons (;) to separate multiple commands on one line
- *swarm* expects bash syntax
- *swarmdel* command (see “*man swarmdel*”)
- Create complex/long swarm command files with scripts

# Demo: Biowulf Applications

Checklist:

1. Is it parallel?
2. Is it multi-threaded?
3. How many independent processes do you need to run?
4. How much memory does one process require?
5. Licensed product? (Matlab)

Biowulf Applications webpages:

<http://biowulf.nih.gov/apps/>

# Demo: R on Biowulf

- Single-threaded
- Large swarms

Demo:

# Interactive Batch Jobs

```
% qsub -l nodes=1 -I [-V]
```

## Reasons to use interactive nodes

- Testing/debugging cpu-intensive code
- Pre/post-processing of data
- Graphical application
- Compiling infiniband application

## Reasons *not* to use interactive nodes

- Easier than setting up a batch job

# Interactive Batch Jobs (2)

- The node is dedicated to the user
- The default allocation is a 2-core, 4 GB node.
- Other nodes may be allocated using a node specification to the `qsub` command.
- The maximum allowable number of interactive jobs is 2.
- By default, the job will be terminated after 8 hours. You may extend this time to a maximum of 36 hours by using the "walltime" attribute to the `qsub` command.

# Parallel jobs

- The original driving force driving compute clusters
- N-communicating processes running on N-cores
- Shared-memory (single-node) vs. distributed-memory (multi-node)
- Message-passing libraries (MPI)
- Communication and scalability
- High-performance interconnects

# High-performance Networks

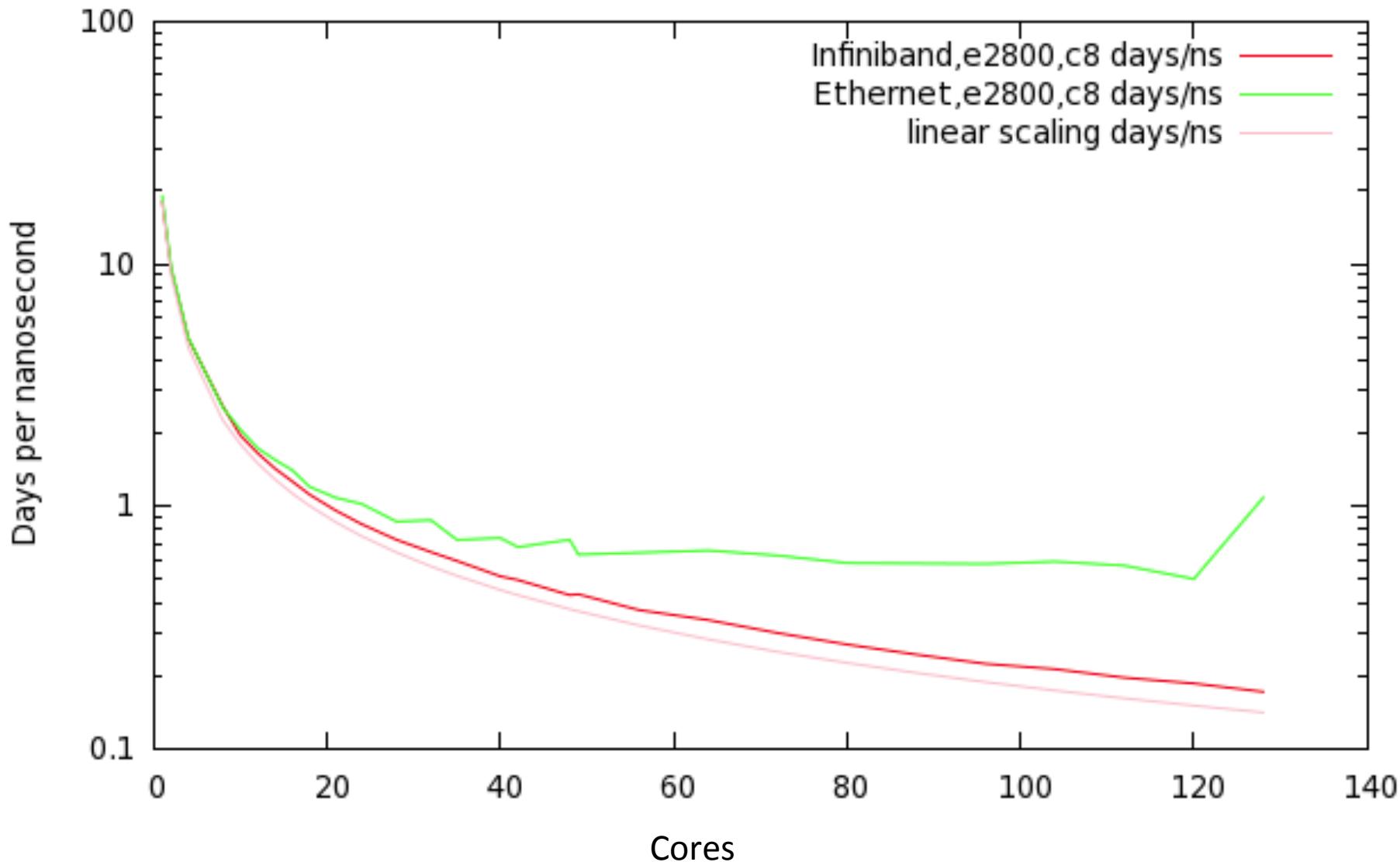
- Parallel (MPI) applications only
- Benchmarking is essential
- Low latency more important than bandwidth
- Bypass TCP/IP stack
- Requires compilation against special libraries
- Two flavors available on Biowulf: Pathscale Infinipath & OFED Infiniband

# Biowulf Cluster Networks

	<b>Gigabit ethernet</b>	<b>Infinipath (Pathscale)</b>	<b>Infiniband</b>
Bandwidth (full duplex)	1 Gb/s	8 Gb/s (SDR)	16 Gb/s (DDR)
Latency	“high”	“low”	“low”
TCP/IP?	Yes	No	No
# nodes	All	132	250
# cores	All	264	2000

# NAMD Benchmarks

APOA1 Benchmark, Infiniband and Ethernet networks for message-passing



# PBS Properties for *qsub* command: node description & selection

Property (resource)...	Selects...
e2666, x2800	2.666/2.8 GHz Xeon processors
o2800, o2600, o2200, o2000	2.8/2.6/2.2/2.0 GHz Opteron processors
c24, c16, c4, c2	24/16/4/2 cores per node
m4, m2, m1	4/2/1 GB per core
g4, g8, g24, g72	4/8/24/72 GB per node
gige	gigabit ethernet
ib, ipath	infiniband, infinipath

# Node Allocation & Selection (qsub)

- Default allocation when using qsub: c2 (o2800 > o2200 > o2000)
- Any other allocation requires selection based on properties:

```
% qsub -l nodes=1:g24 bigmemjob  
% qsub -l nodes=8:o2800:gige namdjob  
% qsub -l nodes=1:c16 tophatjob  
% qsub -l nodes=4:ib -v np=32 namdjob
```

- Use properties only when needed!
- Properties are not orthogonal
- Note: *swarm* uses its own algorithm for node selection (not under user control)

# Available property combinations

Processor	Cores	m1 1 GB mem. per core	m2 2 GB mem. per core	m4 4 GB mem. per core	g4 4 GB total mem.	g8 8 GB total mem.	g24 24 GB total mem.	g72 72 GB total mem.	Network
x2800 2.8 GHz Xeon	c24								
e2666 2.67 GHz Xeon	c16								
o2800:dc 2.8 GHz Opteron	c4								
o2600:dc 2.6 GHz Opteron	c4								
o2800 2.8 GHz Opteron	c2								
o2200 2.2 GHz Opteron	c2								
o2000 2.0 GHz Opteron	c2								
2.8 GHz Xeon	8								ib Infiniband
2.8 GHz Opteron	2								ipath Infinipath

**gige**  
Gigabit  
Ethernet

# Demo: NAMD on Biowulf

- Parallel program using MPI
- Typically low memory requirements (< 1GB/core)
- Benefits from high-performance, low-latency network.

<http://biowulf.nih.gov/apps/namd.html>

Demo

## Node Availability

Free Nodes by memory-per-core					
Cores	m1	m2	m4	Total	
<hr/>					
o2800	2	/	0/204	7/39	7/243
o2200	2	21/299	53/62	/	74/361
o2000	2	29/34	/	/	29/34
<hr/>					
Infiniband					
ib	8	99/250	/	/	99/250
ipath	2	/	11/107	/	11/107
<hr/>					
OnDemand					
e2666	8	113/319	/	/	113/319
gpu2050	12	/	/	9/16	9/16
o2800:dc	4	/	187/224	/	187/224
o2800	2	75/90	/	/	75/90
o2600:dc	4	/	214/250	/	214/250
x2800	24	113/328	/	/	113/328

# Per User Core Limits

- Core limits will vary with system load

```
$ batchlim
Batch system max jobs per user: 4000
Batch system max cores per user: 1280
```

Queue	Max Cores Per User
norm	1280
ib	384
vol	256
norm3	128
g72	128
ipath	128
gpu	96
volib	64

	Max Mem Per User	Max Mem on System
DL-785 mem	512gb	512gb

# 512 GB node

- Shared node
- 32-cores, but single-core jobs using much/all of memory is ok
- To run a job use the “mem=” resource:  
% `qsub -l nodes=1,mem=384gb bigmemjob`
- mem must be >72gb, ≤ 500 gb
- Best not to use “mem=” for any other jobs!
- Assigning multiple cores (default = 1):  
% `qsub -l nodes=1,mem=384gb,ncpus=8 job`

# Back to Swarm

```
$ swarm --help
```

```
Usage: swarm [swarm options] [qsub options]
```

```
-f,--file [file]           name of file with list of commands to execute
-g,--gb-per-process      gb per process
-t,--threads-per-process threads per process. Use "auto" if the application
                           automatically sets the number of threads.

-b,--bundle [ppc]         bundle # processes per core and run sequentially
-R,--resource-list        list of application-dependent resources
                           (e.g. matlab=1,sas=4)

-h,--help                 print this help message
-d,--debug                print debug information and preserve swarm command
                           files

--test                    enable debug mode, no scripts, show mini freen
--usecsh                  use tcsh as the shell instead of bash
```

A selection of PBS qsub options are available (see  
<http://biowulf.nih.gov/apps/swarm.html> for details)

# Swarm bundles

- For large swarms or swarms with short-lived threads (under 5-10 minutes)
- `swarm -b 20 -f commandfile`
- creates a “bundle” (queue) of 20 processes *per core*
- 8000 lines (8000 processes), -b 20 => 25 jobs (16-core)
- Bundles *required* for command files with >10,000 lines

# Swarms of multi-threaded programs

- `swarm -t 8 -v n=8 -f commandfile`
- Number of threads specified to swarm “-t” switch **must** agree with switch specifying number of threads to program (e.g., “-n \$n”)
- Default value of 1

# Specifying memory requirements to swarm

- `swarm -g 4 -f commandfile`
- Swarm will reserve 4 GB of memory *per process*
- Default value of 1
- Swarm will “waste” cores if needed

# swarm --test (1)

```
$ swarm --test -f burn672.swarm  
[test] mini-freen output
```

	NT	Free	/	All	C	M	ppn
	-----						
c2:g1		4	/	80	2	1	1
c2:m1		102	/	666	2	2	2
c2:m2		106	/	532	2	4	2
c2:m4		16	/	78	2	8	2
c4		1512	/	1828	4	8	4
c16		1776	/	5104	16	24	16
c24		2280	/	7872	24	24	24
	-----						
total		5796	/	16160			

# swarm --test (2)

```
$ swarm --test -t 4 -g 8 -f burn672.swarm  
[test] mini-freen output
```

NT	Free	/	All	C	M	ppn
c4	1544	/	1828	4	8	1
c16	1712	/	5104	16	24	3
c24	2592	/	7872	24	24	3
<hr/>						
total	5848	/	14804			

```
$ swarm --test -t 8 -g 8 -f burn672.swarm  
[test] mini-freen output
```

NT	Free	/	All	C	M	ppn
c16	1584	/	5104	16	24	2
c24	2208	/	7872	24	24	3
<hr/>						
total	3792	/	12976			

- --test doesn't submit any jobs

# How To Resubmit a Single Job From a Swarm

```
biowulf% submitinfo 549908
549908: (/gs1/users/susanc/mydir/mysubdir/Rjobs/)
    qsub -l nodes=1:c16 ./swarm/ctrl-swarm6n14942
```

```
biowulf% cd /gs1/users/susanc/mydir/mysubdir/Rjobs/
biowulf% qsub -l nodes=1:g72 ./swarm/ctrl-swarm6n14942
```

## Caveats:

- Do not use ‘swarm’ to resubmit the job. The batch script is not a swarm input file.
- Careful with the number of cores and memory when resubmitting. e.g. if the job ran on a c24 node (24 GB memory) but needs more memory, note that the g72 nodes (72 GB memory) only have 16 cores. You may need to modify the batch script before rerunning.

# Summary: Submitting Jobs to the Batch System

	<i>swarm</i>	<i>qsub</i>
Types of jobs...	<ul style="list-style-type: none"><li>Moderate-to-large numbers of single-node jobs</li></ul>	<ul style="list-style-type: none"><li>Single, small numbers of jobs</li><li>Multi-node (parallel) jobs</li></ul>
Specifications deal with...	application requirements (memory, threads)	hardware (nodes, cores, memory, network)
Job placement on node-types done by...	swarm	User

# Monitoring Batch Jobs

- Things to monitor
  - Core (CPU) utilization
  - Memory
- Tools for monitoring
  - jobload
  - loadmon (web-based)
- *Users* (you) are responsible for monitoring their jobs

# Monitoring Jobs with *jobload*

```
% jobload <jobid>|<user>
```

```
% jobload xxx4
Jobs for xxx4          Node   Cores   Load
2104618.biobos        p41     2       99%
                           p42     2       100%
                           p43     2       99%
                           p45     2       100%
Cores: 8   Job Load Average: 99%
2104620.biobos        p19     2       102%
                           p20     2       99%
                           p21     2       99%
                           p22     2       100%
Cores: 8   Job Load Average: 100%
Core Total: 16      User Load Avg: 100.1%
```

```
% jobload cxx
Jobs for cxx          Node   Cores   Load
2104868.biobos        p1723    4       99%
2104869.biobos        p1726    4       99%
2104870.biobos        p1728    4       99%
.
.
.
2104993.biobos        p1486    4       100%
2104994.biobos        p1487    4       99%
2104995.biobos        p1488    4       99%
Core Total: 512      User Load Avg: 99.9%
```

2106511.biobos	p1666	4	73%
	p1668	4	70%
	p1670	4	67%
	p1672	4	80%
	p1673	4	77%
	p1722	4	83%
Cores: 24	Job Load Average:	75%	

# Poorly behaving jobs

```
% jobload yxxxxu
Jobs for yxxxxu          Node   Cores   Load
2103919.biobos          p79     8        12%
                           p887    8        0%
                           p889    8        0%
                           p890    8        0%
Cores: 32   Job Load Average:      3%
2105929.biobos          p1795   4        50%
2105931.biobos          p1492   4        75%
2105932.biobos          p1493   4        25%
2105933.biobos          p1495   4        50%
2105935.biobos          p1500   4       100%
Core Total: 52           User Load Avg:  34.7%
```

```
# jobload cxxxq
Jobs for cxxxq          Node   Cores   Load
2105432.biobos          p880    8       299%
2105868.biobos          p490    2       100%
Core Total: 10           User Load Avg: 199.9%
```

# Monitoring Memory Usage (1)

```
% jobload -m <jobid>|<user>
```

Jobs for jxxxxxn	Node	Cores	Load	Memory Used/Total (G)
2102742.biobos	p78	16	100%	23.0/74.2
2102743.biobos	p881	16	100%	36.3/74.2
2107026.biobos	p79	16	100%	26.1/74.2

Jobs for jxxxxxxxxs	Node	Cores	Load	Memory Used/Total (G)
1898152.biobos	p1747	4	25%	3.7/8.1
1898156.biobos	p1752	4	25%	3.7/8.1
1898157.biobos	p1753	4	25%	3.7/8.1
1898158.biobos	p1754	4	25%	3.7/8.1
1898162.biobos	p1758	4	25%	3.6/8.1
1898163.biobos	p1759	4	25%	3.7/8.1
1898164.biobos	p1762	4	25%	3.7/8.1
1898168.biobos	p1768	4	25%	3.6/8.1
1898171.biobos	p1772	4	25%	3.7/8.1
1898173.biobos	p1774	4	25%	3.7/8.1
1910428.biobos	p1708	4	25%	3.6/8.2

Jobs for jxxxxxxxxe	Node	Cores	Load	Memory Used/Total (G)
1964089.biobos	p69	16	6%	0.4/74.2
1964090.biobos	p71	16	6%	0.2/74.2
1964091.biobos	p72	16	6%	0.3/74.2

Bad!

# Monitoring Memory Usage (2)

- swap space – disk-based extension of memory
- Swapping (paging) jobs should be killed and resubmitted with larger memory allocations

```
% jobload -m luser
Jobs for luser      Node    Cores   Load      Memory
                                         Used/Total (G)
2206035.biobos     p132      2       142%      1.0/1.0
2206065.biobos     p149      2       136%      1.0/1.0
```

```
% rsh p1841 free -g
              total        used        free        shared      buffers      cached
Mem:          7           7           0           0           0           0           2
-/+ buffers/cache:  5           2
Swap:         3           2           1
```

# Monitoring Memory Usage (3)

- *jobload* gives “instantaneous” usage while a job is running
- What about maximum usage during the lifetime of the job?  
Or the “high-water” mark during a job?

```
% qstat -f 2109191 | grep mem
resources_used.mem = 8392740kb          = 8.4 GB
resources_used.vmem = 8613136kb          = 8.6 GB
```

# Monitoring Memory Usage (4)

After a job has exited...

Email from PBS:

```
From: adm
Date: October 13, 2010 9:39:29 AM EDT
To: user3@biobos.nih.gov
Subject: PBS JOB 2024340.biobos

PBS Job Id: 2024340.biobos
Job Name: memhog2.bat
Execution terminated
Exit_status=0
resources_used.cpupercent=111
resources_used.cput=00:18:32
resources_used.mem=7344768kb = 7.3 GB physical memory
resources_used.ncpus=1
resources_used.vmem=7494560kb = 7.5 GB virtual memory
resources_used.walltime=00:17:05
```

swarm stdout file...

```
----- PBS time and memory report -----
3875594.biobos elapsed time: 236 seconds
3875594.biobos maximum memory: 2.15 GB
-----
```

# Recovering from Deleted Jobs

- Batch system will delete jobs which exceed memory limit for the node
- Fast growing memory may still hang node requiring sysadmin intervention
- jobcheck
- submitinfo

# Demo

- Loadmon

# Demo: Bowtie on Biowulf

- Multi-threaded
- Can require lots of memory

<http://biowulf.nih.gov/apps/bowtie.html>

Demo.

# Demo: Matlab on Biowulf

- Can be multi-threaded.

<http://biowulf.nih.gov/apps/matlab.html>

- License-limited

[http://helixweb.nih.gov/nih/license\\_status](http://helixweb.nih.gov/nih/license_status)

Demo: